

Национальный исследовательский университет «МЭИ»

Институт автоматики и вычислительной техники

**Кафедра вычислительной техники**

**Лабораторная работа №1**

**по курсу** «**Защита информации**»

**Тема: «Защита от несанкционированного доступа к информации»**

|  |  |
| --- | --- |
|  | **Выполнил** |
| **Студент** | Швец Григорий Владиславович |
| **Группа** | А-06-19 |
| **Дата** | 07.03.2023 |

|  |  |
| --- | --- |
|  | **Принял** |
| **Преподаватель** | Андреева Ирина Николаевна |
| **Оценка** |  |
| **Дата** |  |
| **Подпись** |  |

Москва 2023

**Содержание**

[**Цель работы** 3](#_Toc129075515)

[**Задание** 3](#_Toc129075516)

[**Выполнение работы** 3](#_Toc129075517)

[**Исходники** 3](#_Toc129075518)

[**Результат выполнения программы** 5](#_Toc129075519)

[**Проверка полученного результата** 6](#_Toc129075520)

# **Цель работы**

Освоение программных способов идентификации и аутентификации аппаратных средств ПЭВМ с целью защиты от несанкционированного доступа (НДС) к информации.

# **Задание**

1. Составить текст защищаемой программы. Защитить её от НДС.
2. Провести аутентификацию состава аппаратных средств ПЭВМ по вариантам см. табл. 1.

|  |  |
| --- | --- |
| **№ варианта** | **Задание** |
| 17 | Серийный номер материнской платы через WMI |

Таблица 1. Вариант задания.

# **Выполнение работы**

Для реализации поставленной задачи было принято решение использовать объектно-ориентированный язык программирования общего назначения *C#* и среду разработки программного обеспечения *Visual Studio 2022* для написания консольного приложения. Так же была использована сторонняя программа – *WinHex* (универсальный HEX-редактор для Microsoft Windows) для поиска адреса нужной нам строки.

Прежде чем приступить к написанию текста программ стоит дать определение *WMI*. ***WMI*** *(Windows Management Instrumentation)* – это одна из базовых технологий для централизованного управления и слежения за работой различных частей компьютерной инфраструктуры под управлением платформы Windows.

Всего было написано 2 консольных приложения:

1. Защищаемая программа (Prog1.cs)
2. Программа-активатор (Prog2.cs)

# **Исходники**

Замечание (!): для корректной работы программ прежде всего необходимо сделать ссылку на System.Management или подключить аналогичный пакет NuGet. Так же критичным моментом является расположение файлов .exe: программа и активатор должны быть расположены в одной директории.

**Prog1**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Management; // for windows management  using System.Text;  using System.Threading.Tasks;  namespace Prog1  {  class Program  {  static string BoardSerial = "$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$";  static void Main(string[] args)  {  string res\_string="";  // Get serial number of motherboard  string query = "SELECT \* FROM Win32\_BaseBoard";  ManagementObjectSearcher searcher = new ManagementObjectSearcher(query);  foreach (ManagementObject info in searcher.Get())  res\_string =info.GetPropertyValue("SerialNumber").ToString();  // Truncate the length of the BoardSerial string to the length of the received serial number  BoardSerial = BoardSerial.Substring(0, res\_string.Length);  // Check the serial number of motherboard  if (BoardSerial== res\_string)  {  Console.WriteLine("=========== Hello! ===========");  Console.WriteLine("====== Access is allowed ======");  Console.WriteLine("Serial number of motherboard - {0}",res\_string);  } else Console.WriteLine("Attention\nUnauthorized access\n Access denied!\nInvalid serial number.");  Console.ReadLine();  }  }  } |

**Prog2**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.IO;  using System.Linq;  using System.Management; // for windows management  using System.Text;  using System.Threading.Tasks;  namespace Prog2  {  class Program  {  static void Main(string[] args)  {  BinaryWriter bw;  string res\_string = "";  // Get serial number of motherboard  string query = "SELECT \* FROM Win32\_BaseBoard";  ManagementObjectSearcher searcher = new ManagementObjectSearcher(query);  foreach (ManagementObject info in searcher.Get())  res\_string = info.GetPropertyValue("SerialNumber").ToString();  // Activation of the first program  try  {  bw = new BinaryWriter(File.Open(@"Prog1.exe", FileMode.Open));  }  catch (IOException io)  {  Console.WriteLine("File open error " + io.Message);  Console.ReadLine();  return;  }  int seek\_start = 0xA52;  for (int i=0;i< res\_string.Length;i++)  {  bw.Seek(seek\_start, SeekOrigin.Begin);  seek\_start += 2;  bw.Write(res\_string[i]);  }  // Close file  bw.Close();  Console.WriteLine("Activation completed successfully!\n Have a good day!");  Console.ReadLine();  }  }  } |

# **Результат выполнения программы**

**Пример выполнения программы активированного ПК**

|  |
| --- |
|  |

**Пример выполнения программы не активированного ПК**

|  |
| --- |
|  |

**Пример ошибки чтения файла**

|  |
| --- |
|  |

**Пример использования на другом ПК**

|  |
| --- |
|  |

# **Проверка полученного результата**

Для проверки правильности полученного серийного номера материнской платы воспользуемся тем же WMI, только напрямую, через cmd.

Результат:

